

2022 -2023

PROGRAMMATION WEB II : HelbManager2022

Réalisation d'une application web de gestion de projet en Django

RIGGIO JONATHAN & SANCHEZ DOMINGUEZ Angela

Tables des matières

Introduction	2
Description du scénario.....	3
Schéma	5
Fonctionnalités de base	6
Fonctionnalités supplémentaires	11
Difficultés rencontrées	14
Limitations	16
Conclusion.....	17
Bibliographie	18

Introduction

Dans le cadre du cours de programmation Web II, nous devons mettre en place une application web qui permet aux utilisateurs de faire de la gestion de projets via un système simplifié de planification de tâches.

Tout d'abord, pour ce qui est de la réalisation du projet. J'utilise une machine virtuelle Ubuntu disponible sur le réseau de l'école. Dessus j'ai installé l'IDE PyCharm pour pouvoir coder avec le Framework Django. En effet, pour réaliser le projet en Django nous devons coder en plusieurs langages simultanément. En premier lieu, il y a le Python qui est le langage principale de Django pour le back-end. Mais, aussi le HTML, CSS et le JavaScript pour le front-end.

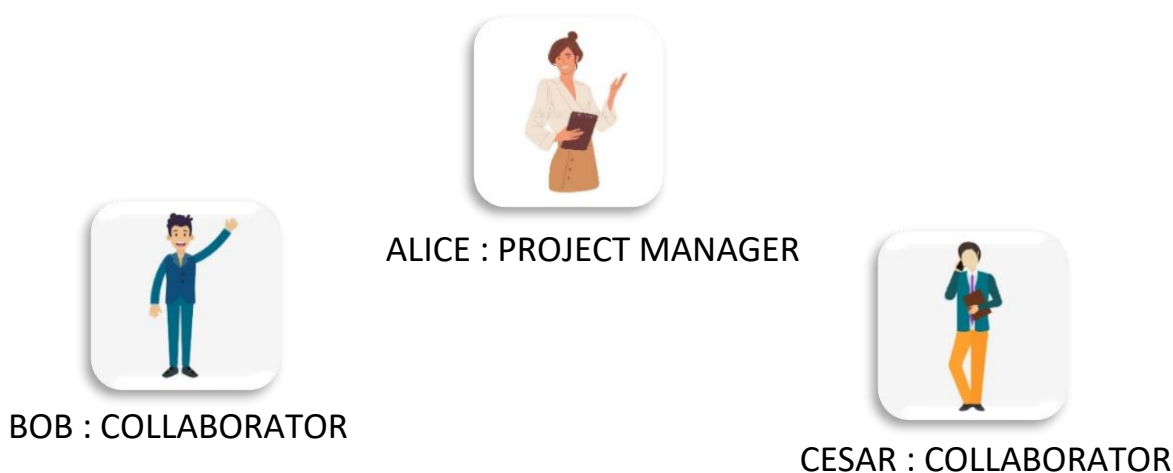
Deuxièmement, le fait de travailler avec un Framework permet de réaliser un site internet facilement. En effet, Django propose des bibliothèques, modules et autres outils. Ce qui rend la structure du site maintenable et évolutive. Bien que la structure de ce Framework soit facile à comprendre, j'ai eu énormément de mal au début.

Car, je n'avais jamais codé encore avec un Framework. Et là passer sur un Framework en plus de ça en Python, j'étais un peu débordé par le nombre d'informations mais grâce à la documentation de Django et quelques vidéos sur YouTube je m'y suis fait.

Sur ce rapport, nous verrons le scénario du projet, son schéma, les fonctionnalités de base, les fonctionnalités supplémentaires, les difficultés rencontrées, les limitations de mon projet. Et finira par la conclusion et la bibliographie.

Bonne lecture :)

Description du scénario



Contexte : Alice est chef d'un projet dont le sujet est « Rapport sur les avantages d'introduire une application mobile ».

Alice crée un nouveau projet sur HelbManager. Donne un titre à ce projet, choisi ses collaborateurs (Bob et César). Pour finir elle met une deadline à son au projet.

HelbManager
Home
About
Contact
Timeline
New Project
Profile
Logout

Project Creation !


Title*

Collaborators*

Deadline*

PS : Collaborators : the format is **Username1,Username2,etc.** No Space but Comma , for Exemple : **'Bob,César'**

PS : For Date, the format is **'Month/Day/Year Hour:Min'** , for Exemple : **'12/31/2022 23:59'**



Welcome Alice !

Après avoir créé le projet Alice créer des tâches et des sous-tâches à son projet et elle y assigne des noms.

Add Task
Show List

Task :

- 1. Rédaction de l'introduction. -Bob
- 2. Recherche de documentation sur les app de vente de produits. -Alice
- 3. Implémentation de l'application mobile. -Bob
- 4. Tests de l'application sur des utilisateurs réels. -César
- 5. Rédaction des résultats des tests. -César
- 6. Rédaction de la conclusion. -Bob

Add SubTask
Show List

Subtask :

- 3.1 Analyse des besoins -Alice
- 3.2 Conception - César
- 3.3 Développement -Bob


Creation Date :

08 January, 2023, 13:14

Deadline :

03 March, 2024, 23:59

Uniquement les personnes faisant partie du projet peuvent accéder au statut du projet (interface Drag & Drop).



Alice
08 January, 2023, 13:14

Update
Delete
Status

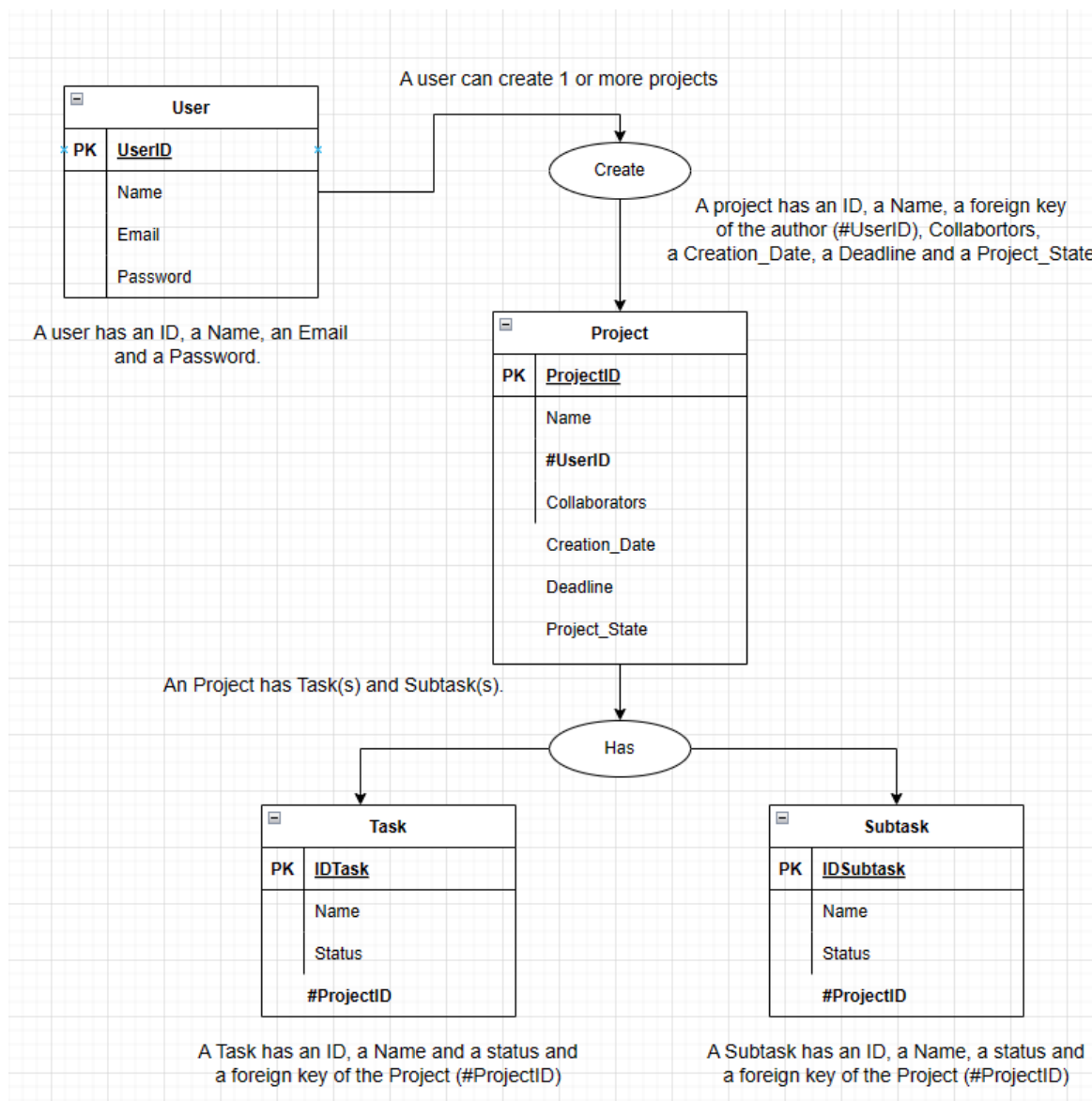
Rapport sur les avantages d'introduire une application mobile

Schéma

Voici la version finale de mon schéma. Celui-ci représente globalement les étapes de mon projet et les relations entre les différentes entités. Notamment on peut noter deux relations importantes.

La première c'est que le Project a la clé primaire de l'User qui devient son l'auteur de ce projet. (#UserID)

La deuxième relation importe c'est la relation entre les tâches et sous tâches et le Project. Les tâches et les sous-tâches lorsqu'ils sont créés ils appartiennent à un projet spécifique. (#ProjectID)

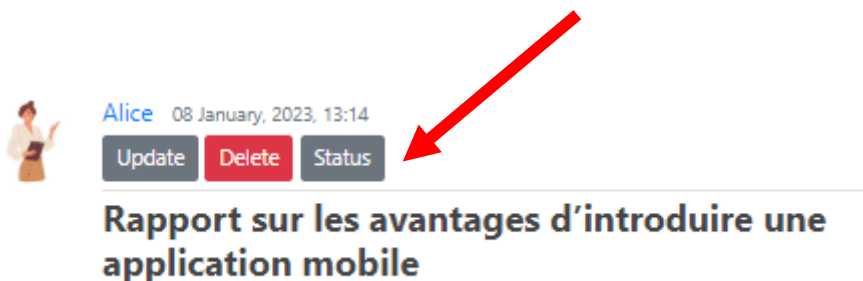


Fonctionnalités de base

1) Une interface de type « drag & drop »

D'abord pour accéder à l'interface drag & drop d'un projet spécifique. Il faut y avoir les droits nécessaires. Soit on est l'auteur du projet, soit on est collaborateurs.

Si on a accès au projet nous pouvons apercevoir le bouton status.



Une fois qu'on accède à la page de l'interface il nous affiche les projets uniquement de ce projet. Ceci est possible à l'aide d'un filter à l'aide d'une clé étrangère (Foreign key).

```
def task_create(request, pk):
    post = get_object_or_404(Post, pk=pk)
    if request.method == 'POST':
```

Dans le views.py on précise une ID comme deuxième paramètre qui correspond à l'ID du projet Post (Projet).

Template :

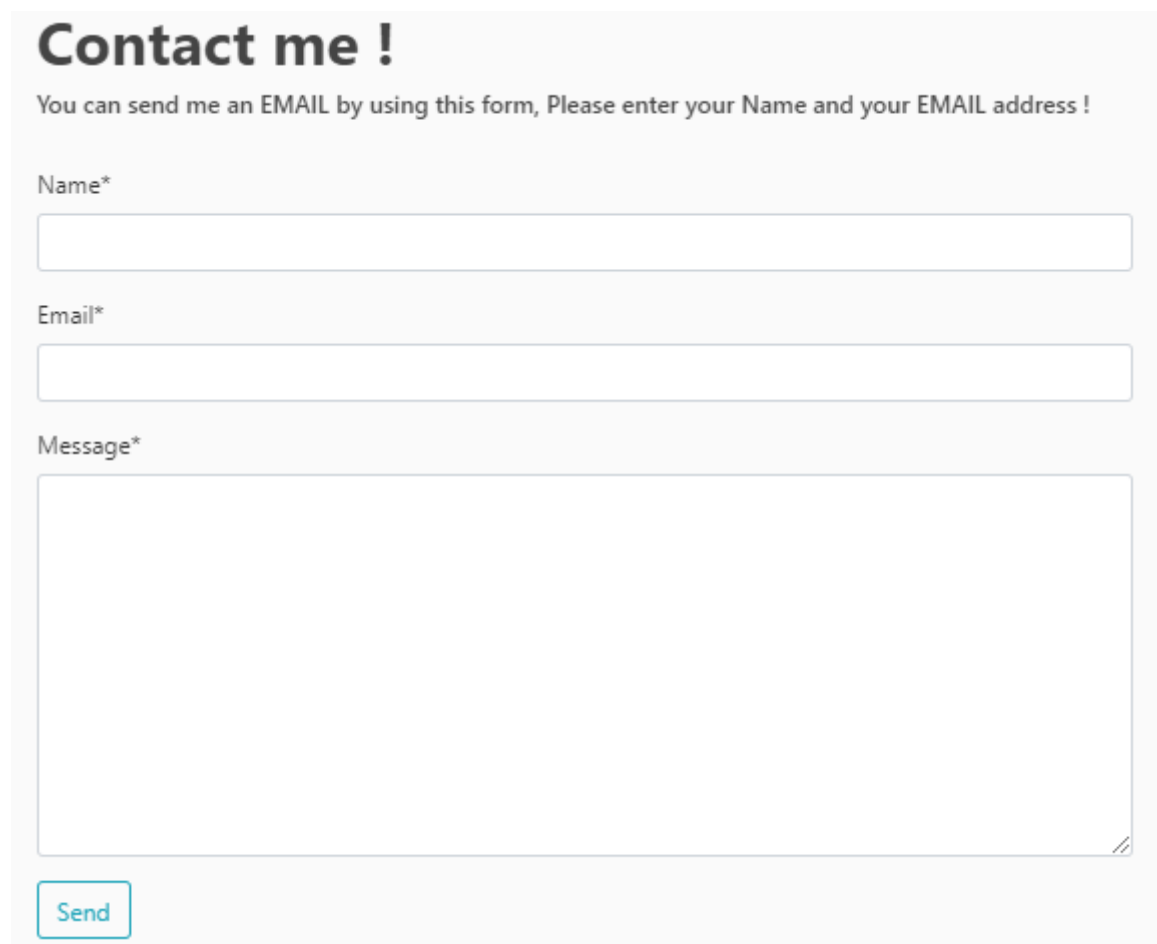
```
<p><b>Task :</b></p>
{% for task in tasks %}
    {% if task.post.id == post.id %}
        <p class="article-content">- {{ task.name }}</pre></p>
    {% endif %}
{% endfor %}
```


2) Application Contact

Pour mon projet j'ai décidé de créer une application supplémentaire qui s'appelle Contact. Pour ce faire il suffit de se diriger dans le répertoire du projet et taper la commande '*Python manage.py startapp Contact*'.

Tout d'abord, j'ai paramétré les répertoires static et les templates. Et puis les fichiers de backend tels que forms.py, views.py pour l'application Contact.

Après j'ai créé un nouveau formulaire en Crispy forms¹ dans mon template pour l'envoi de mail.



Contact me !

You can send me an EMAIL by using this form, Please enter your Name and your EMAIL address !

Name*

Email*

Message*

Send

¹ **Crispy forms** est une bibliothèque de formulaires pour Django qui permet de rendre les formulaires plus attrayants et plus faciles à utiliser en utilisant Bootstrap.

Pour l'envoi de mail en Django il est possible d'utiliser plusieurs services d'envoi de mail. Mais, moi j'ai décidé de travailler avec Mailtrap.io.

Dans le settings.py de mon projet j'ai mis les paramètres spécifiques à Mailtrap.io de mail

```
# SETTINGS FOR MAILTRAP
EMAIL_HOST = 'smtp.mailtrap.io'
EMAIL_HOST_USER = 'fdfa280a63570f'
EMAIL_HOST_PASSWORD = 'b3b9e9e206be23'
EMAIL_PORT = '2525'
```

Dans mon fichier views.py de mon application Contact j'ai dû importer

```
from django.core.mail import send_mail
```

Pour l'envoi de messages, on n'envoie pas de String ou quoi que ce soit. On envoie directement une page HTML. Voici le contenu du fichier HTML que j'envoi par mail. Pour plus d'informations il y'a le site zetcode.com qui parle plus en détails sur les différentes étapes de configuration.

```
<h2>New mail received !</h2>

<p><b>Name : </b>{{ name }}</p>
<p><b>E-mail : </b>{{ email }}</p>
<p><b>Message : </b><pre>{{ message }}</pre></p></pre>
```

New mail received !

Name : Alice

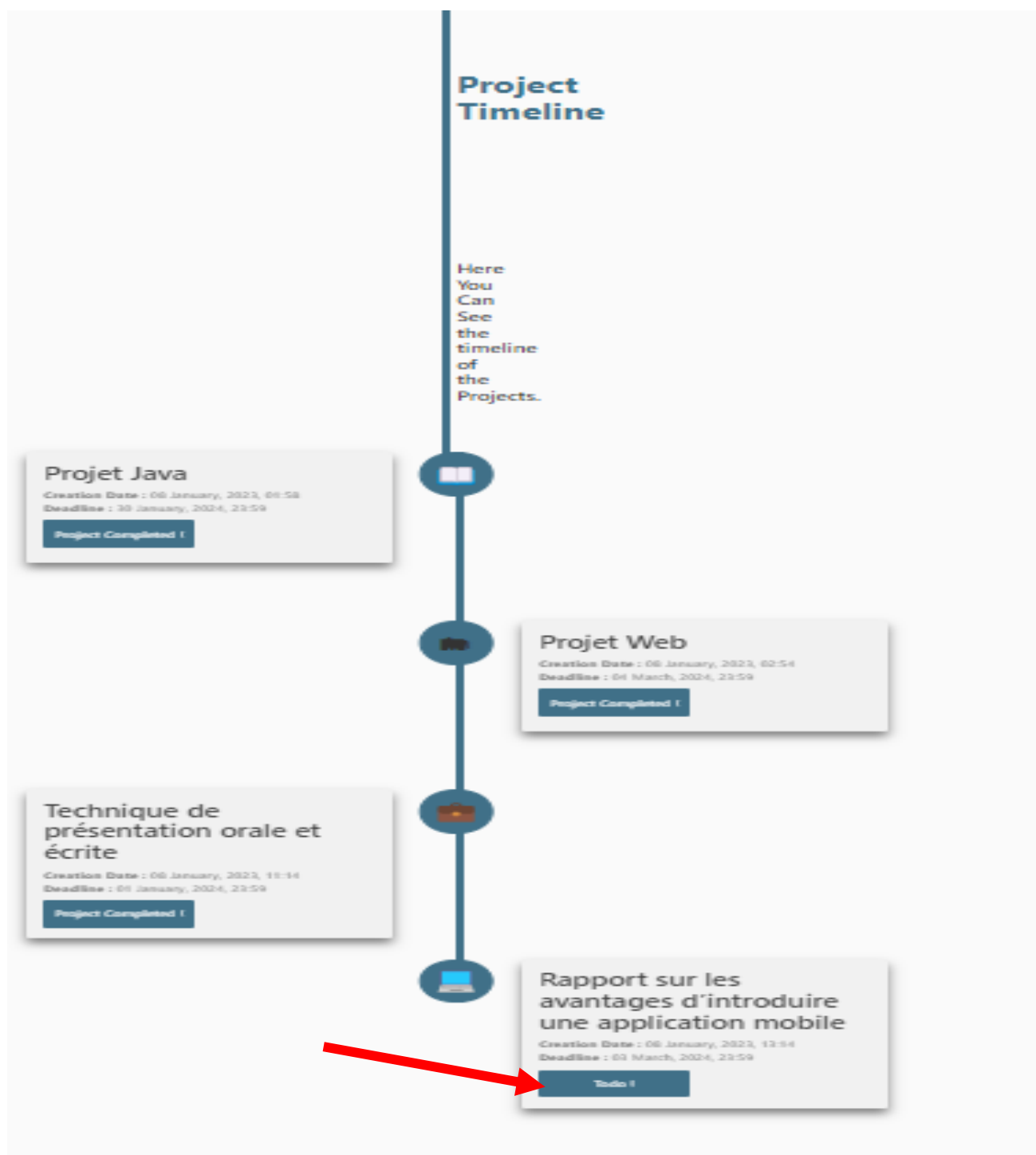
E-mail : Alice@gmail.com

Message :

Task 2 : In_Progress
Task 3 : Done

2) Application Timeline

Pour ce qui concerne le timeline qui permet de visualiser graphiquement l'état d'avancement du projet, j'ai décidé de procéder un peu différemment. Plutôt que de mettre les tâches et les sous-tâches. Je mets directement les projets. Si un projet n'est pas fini on peut appuyer sur un bouton et ça nous dirigera directement vers l'interface drag & drop du projet en status **TODO**.



Fonctionnalités supplémentaires

1) Petite mesure de sécurité (intégrité des données pour les utilisateurs)

Les informations ne pourront plus être modifiées depuis l'interface Admin. Uniquement le chef de projet peut modifier le contenu de son projet. C'est une petite mesure de sécurité mais nous ne sommes jamais sûrs de ce qui peut arriver.

Par exemple : Une personne qui aurait accès illégalement à l'interface Admin pouvait changer les informations concernant les projets. Par exemple, changer la deadline, changer le contenu des projets, retirer des collaborateurs, etc.

Pour ce faire nous passons toutes les informations dans l'interface Admin en readOnly, il ne pourra que lire les informations mais plus modifier le contenu.

J'ajoute à l'interface admin les informations sur les Projets, Taches et sous-tâches.

```
admin.site.register(Post, PostAdmin)
admin.site.register(Task, TaskAdmin)
admin.site.register(Subtask, SubtaskAdmin)
```

Mais je les paramètre en lecture seule. Pour l'intégrité des données.

```
class PostAdmin(admin.ModelAdmin):
    readonly_fields = ('title', 'date_posted', 'deadline', 'author', 'collaborators')

class TaskAdmin(admin.ModelAdmin):
    readonly_fields = ('name', 'status')

class SubtaskAdmin(admin.ModelAdmin):
    readonly_fields = ('name', 'status')
```

2) Librairie d'animation au scroll (AOS.JS)

Comme fonctionnalités supplémentaires j'ai également décidé d'ajouter des effets visuels à mon site internet. En effet, JavaScript propose énormément d'effets visuels. J'ai décidé de travailler avec la librairie AOS.JS (Animation On Scroll) qui permet d'ajouter directement des effets aux éléments HTML. sans devoir coder dans un fichier JavaScript
La procédure est très simple. Il y a 3 étapes.

1) J'ai lié mon Base.Html aux deux liens CSS et JS.

```
<!-- AOS ANIMATION ON SCROLL LIBRARIES (CSS) -->
<link href="https://unpkg.com/aos@2.3.1/dist/aos.css" rel="stylesheet">

<!-- AOS ANIMATION ON SCROLL LIBRARIES (JAVASCRIPT) -->
<script src="https://unpkg.com/aos@2.3.1/dist/aos.js"></script>
```

2) Mettre ce petit script à la fin du body du fichier Base.html qui va permettre de déclencher l'animation

```
<!-- TRIGGER ME SCROLL ANIMATION ! -->
<script>
  AOS.init();
</script>
```

3) Et pour finir il faut tout simplement créer des '<div data-aos="">' »

```
<!-- DRAG AND DROP INTERFACE -->
<div data-aos="fade-up" data-aos-duration="3000">
```

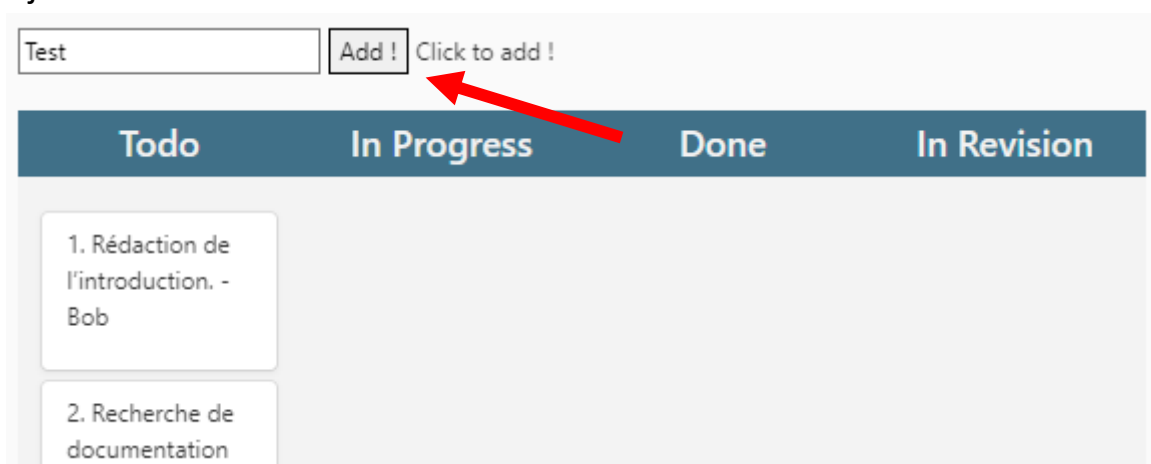
Cette ligne de code permet de mettre un effet visuel qui se déplace vers le haut en 3 secondes. Il y a beaucoup d'autres options pour plus d'informations il est possible d'aller sur le site internet d'AOS.JS. (Source en bibliographie).

3) Possibilité d'ajouter des tâches et de supprimer directement dans l'interface drag & drop (BONUS)

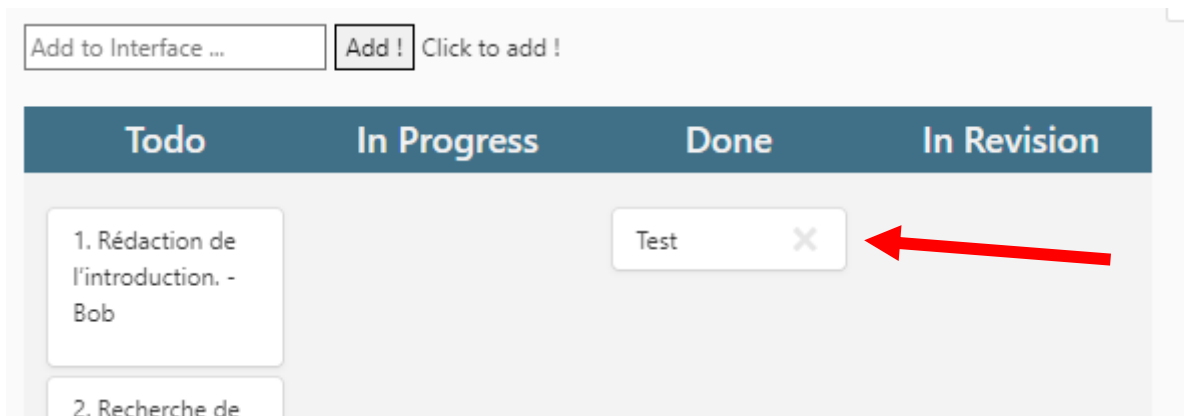
Lorsque j'effectuais mes recherches sur l'interface drag & drop, je suis tombé sur la vidéo YouTube de Basir Payenda qui permettait d'ajouter et de retirer des éléments dans l'interface de drag & drop.

Évidemment ce n'est que du bonus, ça n'apporte rien de concret au projet. Car, les Task se trouvent déjà dans l'interface. Les fichiers CSS, et JavaScript ont été inspirés de la vidéo de Basir Payenda. (Source en bibliographie).

Ajouter éléments :



Supprimer éléments :



Difficultés rencontrées

- 1) Pour la création du projet (new Project) sur mon site j'avais du mal à ajouter des champs supplémentaires. En effet, le POST ne fonctionnait pas. J'ai donc tapé sur Google l'erreur. Et je me suis rappelé que je devais mettre à jour la base de données. En créant une migration et en l'exécutant.

Makemigrations : Créer, générer les commandes SQL.

Migrate : Exécuter les commandes SQL.

- 2) Problème lors de la création d'un nouvel utilisateur.



CODE :

```
def save(self):
    super().save()
```

Dans le "users/models.py" :

J'ai surchargé la méthode save(), mais je dois mettre je n'avais pas préservé sa signature. Je dois envoyer les mêmes arguments que la méthode originale, et les transmettre lors de l'appel à super.

*args et **kwargs permettent de récupérer les arguments sous forme de dictionnaire.

```
def save(self, *args, **kwargs):
    super(Profile, self).save(*args, **kwargs)
```

3) Foreign key

Pour la création du Foreign key de mes tâches et sous-tâches j'ai eu énormément de mal. J'ai dû supprimer toute ma base de données. Car, j'avais une incohérence de clé étrangère.

'**Python manage.py flush**' pour supprimer toutes les données de la base de données.

Après j'ai recherché comment je pouvais visualiser ma base de données pour avoir une vue d'ensemble et j'ai trouvé sur internet la méthode pour afficher directement la base de données Django sur Visual Studio Code.

Et grâce à ça j'ai pu mieux visualiser et comprendre les relations entre les différentes entités de mon projet.

4) Pour l'envoi d'email j'ai eu aussi énormément de mal. Car, je n'arrivais pas à comprendre pourquoi je ne pouvais pas envoyer de message. Puis j'ai effectué une recherche et je suis tombé sur le site **de zetcode.com** .En effet, **il fallait envoyer directement des pages html et non des messages normaux.**

Limitations

- 1) Le plus gros des limitations selon moi, c'est le fait que le drag & drop ne garde pas son état. A chaque refresh les informations reviennent à leur place initiale.
- 2) Dans le drag & drop, les tâches assignés aux utilisateurs ne sont pas accessibles par n'importe quels utilisateurs du site. Mais, le problème est que Bob par exemple : peut bouger la tâche de César dans l'interface drag & drop.
- 3) J'aurais pu faire un système de message en temps réel directement intégrée à mon projet. Mais j'ai décidé de travailler avec un système de mail.

Conclusion

Pour conclure mon projet, je dirai que c'était une expérience très enrichissante. Le fait d'avoir un projet à rendre pour une date limite m'a permis de m'améliorer sur énormément de points. En ce qui concerne l'organisation de mon travail ou alors pour ma méthode de travailler. En effet à force de rechercher sur Google, ou alors lire la documentation Django. J'ai compris énormément de chose sur le fonctionnement de Django.

D'ailleurs la réalisation de ce projet m'a permis de me re familiariser avec python dont je n'avais plus touché depuis deux ans.

Au début, j'étais vraiment débordé par le nombre d'informations mais j'ai su m'y faire à l'aide des vidéos explicatives disponibles sur YouTube. Durant la réalisation j'ai rencontré aussi pas mal de problèmes que j'ai pu surmonter à l'aide de Stackoverflow, w3schools, etc.

Si j'avais eu plus de temps j'aurais fait en sorte d'enregistrer en temps réel les status de mes tâches directement dans ma base de données.

Ce projet était très plaisant à réaliser malgré les moments de difficultés.

Bibliographie

- 1) Corey Schafer. (2018, août 31). *Python Django Tutorial : Full-Featured Web App Part 1 - Getting Started*. YouTube. <https://www.youtube.com/watch?v=UmljXZlYpDc>
- 2) W3schools, HTML Drag and Drop API. (s. d.). https://www.w3schools.com/html/html5_draganddrop.asp
- 3) AR-M-AN, Stackoverflow - save() got an unexpected keyword argument « force insert ». (2018, 16 septembre). Stack Overflow. <https://stackoverflow.com/questions/52351756/django-typeerror-save-got-an-unexpected-keyword-argument-force-insert>
- 4) AOS - Animate on scroll library. (s. d.). <https://michalsnik.github.io/aos/>
- 5) Basir Payenda. (2020, 21 December). *To Do App Using HTML, CSS and JavaScript (Drag & Drop)/Project # 10/100* [Video]. YouTube. <https://www.youtube.com/watch?v=m3StLl-H4CY>
- 6) Bodnar, J. (s. d.). *Django email - how to send emails in Django*. <https://zetcode.com/django/email/>
- 7) Django. (s. d.). Django Project. <https://docs.djangoproject.com/en/4.1/ref/templates/builtins/>
- 8) *Django FR*. (s. d.). [Vidéo]. YouTube. https://www.youtube.com/playlist?list=PLC33I5VK9ILVKw0XG1ZrCL_Q-9SLacCTN
- 9) Team, M. (2023a, Janvier 5). *Mailtrap : Email Delivery Platform*. Mailtrap. https://mailtrap.io/?gclid=CjwKCAiA2L-dBhACEiwAu8Q9YCwCPNoPQIhmYrKofuhtExhY18DytcscsB8zLMb7RP-wwYYHwuoyUhoClXsQA_vD_BwE